

I-D-S

Introduction To Integrated Data Store



GENERAL  ELECTRIC

INTRODUCTION TO INTEGRATED DATA STORE

April 1965

GENERAL  ELECTRIC

COMPUTER DEPARTMENT

PREFACE

This manual presents a general description of the IDS system and discusses its features and capabilities. A description of the source language is also presented; however, detailed programming information is not discussed. Programming details and techniques will appear in a forthcoming IDS reference manual.

The IDS system is mass memory oriented; therefore some information on disc storage units is presented. However, for those who require more detailed information, refer to the DS-20 Disc Storage Unit reference manual, CPB-345 and forthcoming manuals on DS-15 and DS-25.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2691, Phoenix, Arizona, 85002.

© 1965 by General Electric Company

COMPATIBLES

IDS

CONTENTS

	Page
 1. INTRODUCTION	
The Need for IDS	1
Advantages of IDS	1
Data Organization Techniques	2
Record Processing Language	2
The Computer System	4
 2. INTEGRATED SYSTEMS DESIGN	
Comparison of Systems Design Approaches	5
Conventional File Organization	5
IDS File Organization	7
Associating Records into Chains	8
 3. IDS ENVIRONMENT	
Disc Storage	11
DS-20	11
Read/Write System	12
IDS Pages	13
Reference Code	14
Input/Output Controller	14
Data Buffers	14
Priority Control	14
Record Unpacking	15
File Protection	15
 4. ELEMENTS OF IDS	
Data Organization	17
Data Records and Fields	18
Record Classes	19
IDS Chains	19
Multiple Chains	20
Chain Processing	22
Linking a New Detail Record	22
Master Record Selection	23
Chain Ordering	23
Prime Chain	23
Data Structure	24
Summary of Data Structure	26

	Page
5. RECORD PROCESSING LANGUAGE	
Source Language	29
Procedural Statements	31
Imperative Statements	34
Conditional Statements	38
File and Data Descriptions	39
Record Description	40
Chain Definition	44

ILLUSTRATIONS

Figure		Page
1.	Compiler Processing Functions	3
2.	Retrieval Function	3
3.	Conventional Record Format	6
4.	DSU Layout--Conventional	6
5.	Record Access	7
6.	IDS Record Formats	7
7.	Chaining Example	9
8.	Disc Surface Configuration	11
9.	Conventional Disc Records	12
10.	IDS Page	13
11.	Input/Output Controller	15
12.	IDS Records	17
13.	Chain Association	18
14.	IDS Chains	20
15.	Master Record of Two Chains	21
16.	Chain Processing	22
17.	IDS Shorthand	24
18.	Purchase Order Data Structure	25
19.	Chain Network	26
20.	Legal IDS Structures	27
21.	Illegal IDS Structure	27

1. INTRODUCTION

Integrated Data Store (IDS) is a new information-oriented method of integrating the operating functions of a business. It allows you to sharply reduce the high systems and programming costs associated with implementing an integrated business system. As a general-purpose system, it uses mass random access storage as an extension of memory and gives you an efficient data organization technique and a simple but effective language to operate the system.

THE NEED FOR IDS

Integrated business systems which operate with mass random access storage are considerably larger and more complex than systems for conventional card or tape processing. Complex relationships of business data must be structured in a way that is meaningful to the user and yet conforms to particular characteristics of mass random access processing. Before IDS, this proved to be an overwhelming task involving 18 months to 2 years before the system was truly operational.

ADVANTAGES OF IDS

The IDS software system offers many advantages in the design and programming of information systems. Some of these advantages are:

- Shortened Time Span--Systems design and programming involved in the implementation of business systems are greatly reduced.

Systems Design--A simplified, disciplined approach is provided for documenting the highly interrelated data structure inherent in most business systems.

Record File Layout and Organization--The IDS system provides a unique technique for organizing records based on their meaning and association with related records in a mass storage environment.

Programming--IDS macro-instructions reduce programming for record storage, retrieval, and processing.

- Improved Communications--A disciplined documentation approach is provided which enforces the consistent application of sound principles.

Business Systems--IDS provides an effective communication tool between systems designers and programmers.

Management--Structuring of the total business system is documented in a form that management can understand.

COMPATIBLES

IDS

- Reduction of Costs--The extensive costs involved in the design and programming of complex systems are materially reduced.

Personnel Training--Time and cost involved in training personnel in hardware characteristics and file organization techniques is reduced when using IDS.

Programming--Programmers can use their time more efficiently by focusing their attention upon the application logic and results rather than the maintenance of records (storage and retrieval).

Program Checking and Debugging--IDS macro-instructions carry out gross functions and any programming errors that may occur would consist of selecting the wrong function. Errors of this type are immediately noticeable.

Conversion--The compatibility features of IDS reduce conversion costs as installations outgrow present equipment or expand their system.

- More Efficient Use of Storage Capacity--IDS automatically provides for maximum utilization of the mass storage device.

Redundant Data--The design of IDS allows the elimination of redundant data from the file records.

Consolidation of Records--IDS automatically organizes and packs records within the mass storage, providing maximum utilization.

- Reduction of Record Maintenance--The time required for record updating is less than for present systems, since information is carried only once in the IDS system. This also increases accuracy as there is only a single source of information to maintain and all reports produced from that source will reflect all changes.

DATA ORGANIZATION TECHNIQUES

IDS provides a convenient method of describing complex information structures through the meaningful association of the data record contents. Once the data is described, the system automatically structures it to suit the hardware requirements of the mass storage device. The task of organizing data records for meaningful association is handled by the IDS system. This record association is achieved through the use of chains, which provide cross-reference linkages between records. These record chains provide the integrating force of IDS.

RECORD PROCESSING LANGUAGE

The IDS language provides a simplified means for record processing in the environment of mass random access storage.

Present procedural languages offer programming convenience in field and sequential record processing. However, they are inadequate when processing records in the random environment of mass storage. Language statements such as those for read/write operations produce serial rather than random actions. The burden of organizing data records and the logic involved in processing and maintaining these records is placed upon the programmer. These same functions are performed automatically by the IDS software system.

The IDS language extends the range of the normal compiler language beyond the base of COBOL. Four powerful IDS instructions--STORE, RETRIEVE, MODIFY and DELETE--perform the logical functions of record storage, retrieval, modification and deletion. These macro-instructions work in conjunction with, and supplement, the normal COBOL language, as illustrated in Figure 1.

	PROCESSING Function			Data Description
	Procedure	Field	Record	
C O B O L	GO TO PERFORM	MOVE ADD SUBTRACT etc.	READ WRITE	Field Record File
I D S		MODIFY	STORE RETRIEVE DELETE	Field Record Chain

Figure 1. Compiler Processing Functions

The IDS language gives COBOL a powerful mass random access capability while maintaining the full use of the field manipulation characteristics of COBOL. The power of joining COBOL and IDS languages is shown by the logic capability inherent in the RETRIEVE macro-instruction, as illustrated in Figure 2.

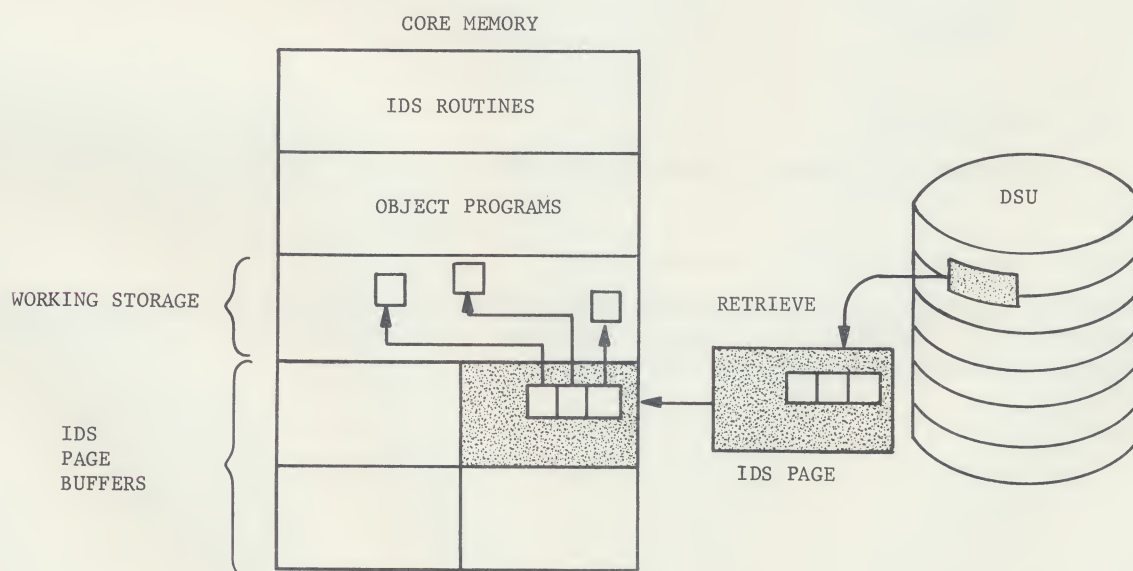


Figure 2. Retrieval Function

The RETRIEVE macro-instruction retrieves a record from the disc storage unit (DSU) and makes the data content available in computer memory for the problem logic programming by COBOL. At the same time the data record itself is protected from accidental destruction.

THE COMPUTER SYSTEM

IDS is available for use with any member of the General Electric Compatibles/200*/400/600 families of computers, when the system includes a disc storage unit. The development philosophy of IDS was such that it can be implemented on future mass storage devices as they become available.

The minimum hardware configuration needed by the IDS system for each type of General Electric computer system is shown below:

The Compatibles/200

- 8192 Words of memory
- 1 Disc storage unit
- 1 Card reader
- 1 Card punch
- 1 Printer
- 1 Typewriter

The Compatibles/400

- 16384 Words of memory
- 1 Disc storage unit
- 4 Magnetic tape units
- 1 Card reader
- 1 Card punch
- 1 Printer

The Compatibles/600

Any system which includes a disc storage unit.

*The specific language (COBOL implementation) outlined in this manual does not apply to Compatibles/200 IDS. However, the language functions are available in the Compatibles/200 implementation and have been in use since early 1964.

2. INTEGRATED SYSTEMS DESIGN

The design of integrated data systems is highly complex. The conventional system is organized function-by-function, each with a bulky file. Mandatory cross-referencing to show interaction between functional operations becomes highly intricate and in many cases necessitates carrying redundant information within the files. Many computer runs are necessary to process these files. Then, the results must be coordinated and correlated. IDS provides an answer to these problems: a single file organized around the data rather than the function.

The concept of Integrated Data Store significantly simplifies the design of integrated systems by permitting the establishment of meaningful associations between data records without any redundancy. A comparison between the conventional approach and the IDS approach to systems design in the following discussion illustrates the benefits of the latter.

COMPARISON OF SYSTEMS DESIGN APPROACHES

Conventional File Organization

When designing an information system, the designer must resolve many questions. Some of these are:

- What information must be stored in the system?
- How should this information be stored (volume, sequence, accessibility, etc.)?
- What cross references must exist between information in the various files?
- How will information be retrieved in response to information processing requirements?

If, in a conventional approach to file organization, the problem being studied is the design of a purchasing information system, it may be necessary to plan three duplicate files of purchase orders:

1. A vendor file with related open purchase orders for general processing.
2. A purchase order file for expediting purposes.
3. A file by inventory item number for production inquiries.

Figure 3 illustrates a conventional approach to file organization using disc storage.

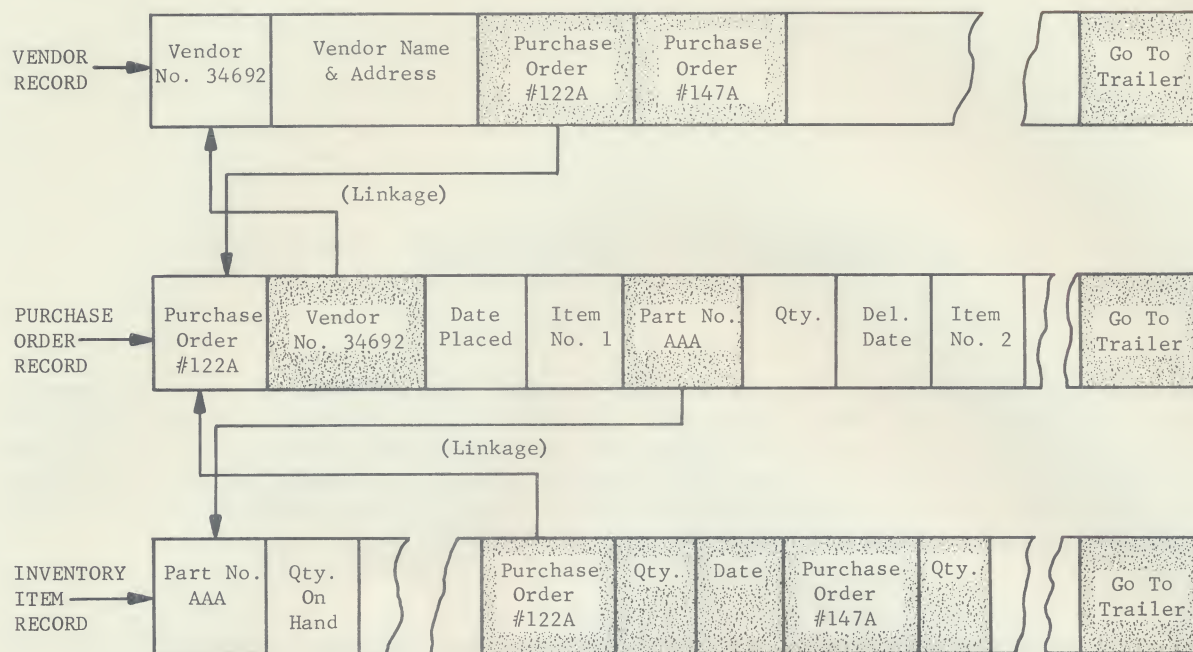


Figure 3. Conventional Record Formats

You will note that the series of records shown in Figure 3 contains a considerable amount of redundant data (shaded fields). These records are typically stored in separate areas of the file unit with trailers to separate overflow areas, as shown in Figure 4. (Overflow areas are an extension of the normal file area. They store the records with the duplicate disc addresses often created when using randomizing techniques for record storage.)

The systems design and programming effort involved in designing the file layout record content and data associations, to take full advantage of the hardware capabilities, is both complex and lengthy.

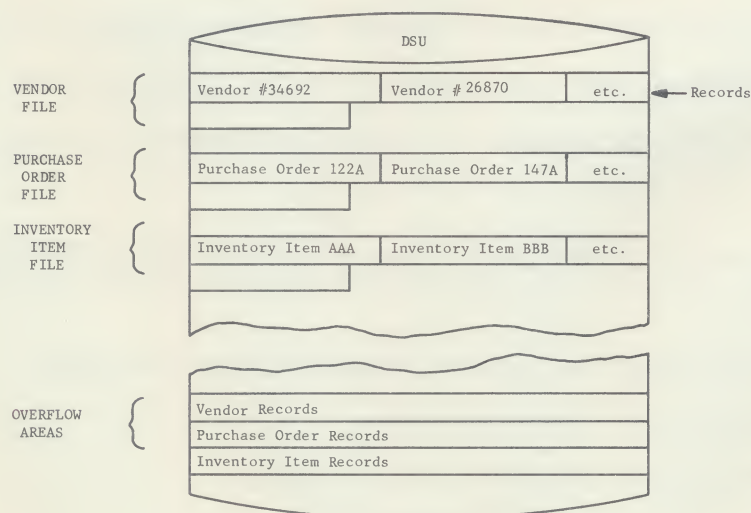


Figure 4. DSU Layout--Conventional

The processing and maintenance of records stored, as already shown, is both cumbersome and time-consuming. Related information is packed in different sections of the file unit, and records must be individually accessed in each location to complete the maintenance function (Figure 5).

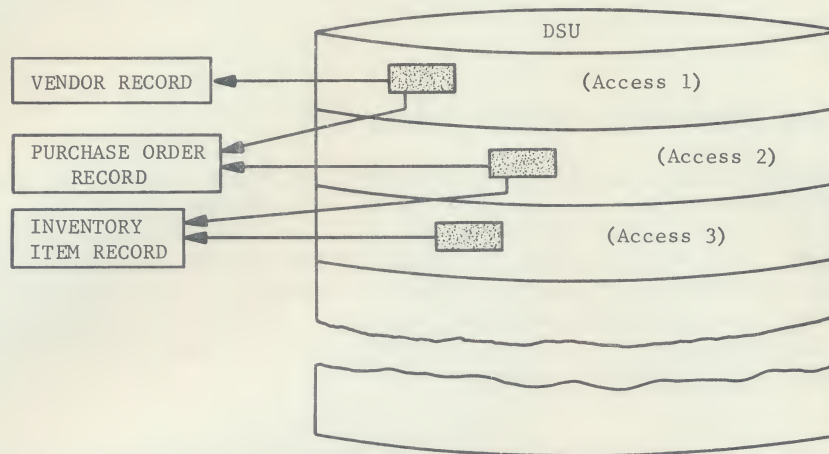


Figure 5. Record Access

If there can be several hundred or several thousand open purchase orders at any one time and the activity (changes, new orders, receipts, etc.) is high, then maintaining these files require considerable processing time.

IDS File Organization

IDS was designed to relieve the systems programming and storage problems inherent in the conventional approach to data organization and the subsequent processing of this data.

The IDS record construction comparable to the examples of conventional file organization appear in Figure 6.

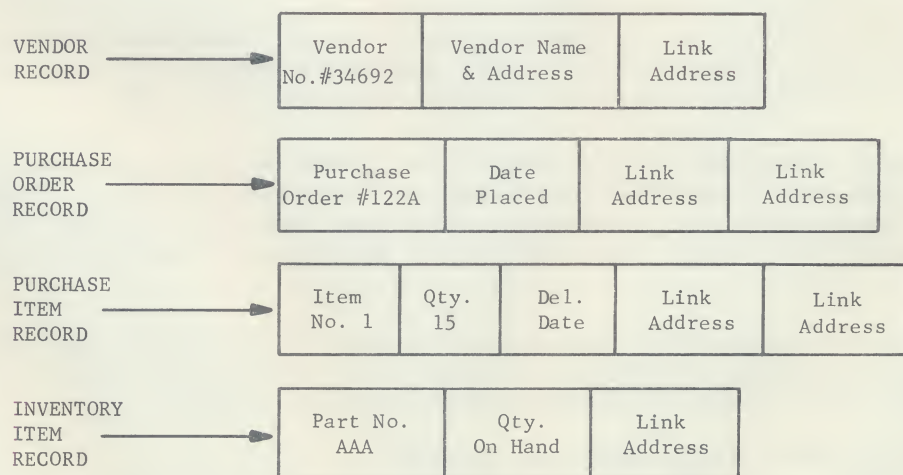


Figure 6. IDS Record Formats

You will note there is no redundant information in the above records. Therefore, the purchase order record, to convey its full meaning, must be properly associated with (1) the vendor record which describes the vendor upon whom the order was placed (2) the item records which describe the quantity ordered and the delivery data, and (3) the inventory records which described the items being purchased.

ASSOCIATING RECORDS INTO CHAINS

The chaining feature is the fundamental structuring tool of IDS. Chains are made up of all the information about a particular function, as in the vendor record in Figure 7. A chain must contain one master record (the vendor record) and can contain any number of detail records (other data directly related to the vendor). All the interlocking relationships of pertinent information are cross-referenced by the chains. Chains are linked together automatically. A master record in one chain may be a detail record in another. There is no redundancy in the storing, processing, or maintaining of data. Figure 7 illustrates the chaining network of logical records as they might appear in an information system.

With an information system structured in this manner, file interrogation and processing is conveniently simplified.

In examining the records and chaining associations in Figure 7, note that there are four types of records:

- Vendor
- Purchase order
- Item
- Inventory

To provide meaning to the system, these logical records are associated in chains:

- All the purchase order records--for a vendor
- All the item records--for a purchase order
- All the order item records--for an inventory item

Within each chain there can be a variable number of records. These records can be linked into any number of chains. The information system organized in this manner permits interrogation by all functions of the business with records and data stored only once in IDS.

In the IDS system, it is possible to store related records within the same block on the disc storage unit. During each disc access a block of information is transferred to memory. Therefore, with proper data organization, the probability is high that, with a single access to the disc for a particular record, most of the related information will also be available in the block stored in memory.

With this feature in mind and by using Figure 7, it is easy to show how the purchasing function of the business can obtain all information pertinent to vendor status. For example, if information is requested for vendor number 34692, the processing sequence of Figure 7 is:

1. Retrieve vendor record (vendor code is 34692).
The basic vendor data is now available in memory for processing. (In each vendor record there is a chain link to the first purchase order record.)

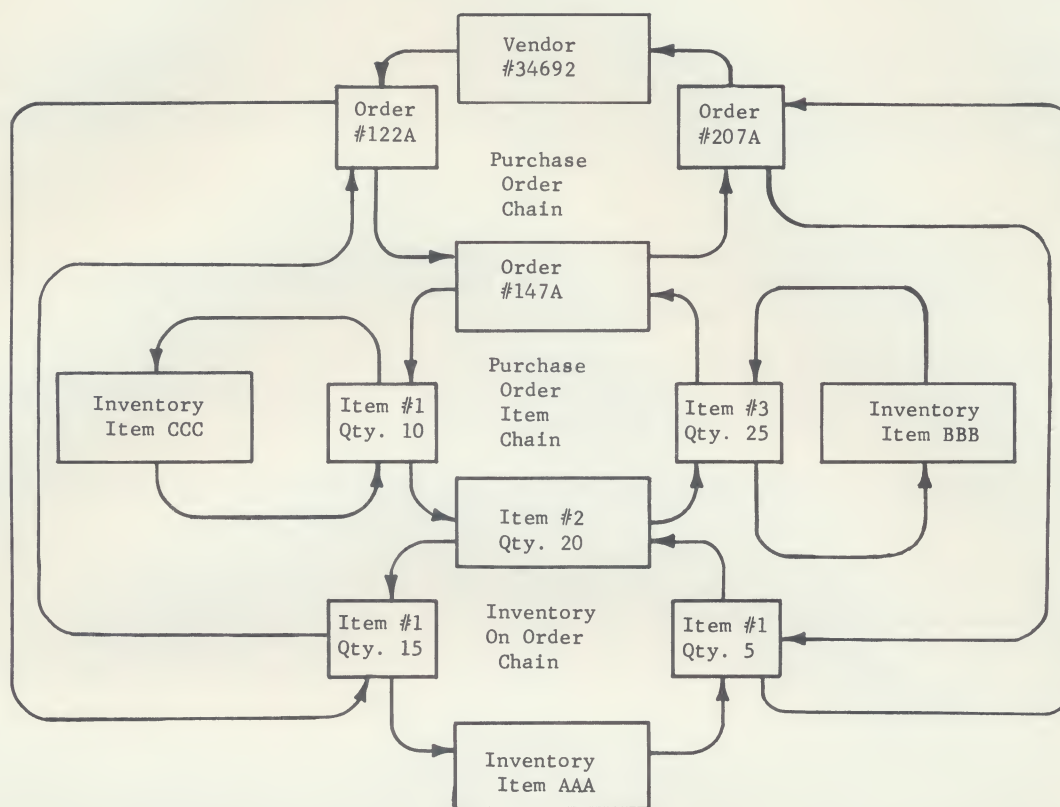


Figure 7. Chaining Example

2. Retrieve next purchase order record in purchase order chain.
Process data of purchase order number 122A. (In each purchase order record there is a chain link to the first item record in the item chain and a link to the next purchase order record in the purchase order chain.)
3. Retrieve next item record in order item chain.
Process data of item number 1. (The last item record contains a chain link back to the purchase order chain.)
4. Processing of purchase order 147A with its item records 1, 2 and 3 occurs in the same manner.
5. Following processing of purchase order number 207A, the purchase order chain links back to the vendor record, which completes the cycle for this vendor.

Composite vendor information is stored and retrieved through these chains, which provide a meaningful association of related data. Redundancy is eliminated; for example, vendor number is carried only in the vendor record, which is then associated with its purchase orders through chains.

Flexibility of the IDS system organization can best be illustrated by viewing a second method of processing the records in Figure 7. The production control functions of the business might, for example, inquire as to the inventory status, both on hand and scheduled on order, of inventory item AAA. The processing sequence is as follows:

1. Retrieve inventory item (inventory item number is AAA).
Process the inventory on hand balance. (Each inventory record contains a chain link to the first item in the on order chain.)
2. Retrieve next item record in the inventory on order chain.
Process item number 1 (quantity 5). (NOTE: The item record is in two chains--the inventory on order chain and the purchase order item chain.)
3. Repeat step 2 until the last item record in the inventory on order chain is processed and the chain terminates back at inventory item AAA.

When the inventory status indicates the need for vendor expediting of an order item, this can be accomplished simultaneously with the above processing. This is possible because the items in the inventory on order chain are also chained into the order item chain and in turn into the purchase order chain. By traversing these chains, all necessary purchase order and vendor data can be obtained.

3. IDS ENVIRONMENT

For the reader to understand the overall organization and concept of IDS, general knowledge of a disc storage unit (DSU) is required. In the following discussion, some basic information on the DSU is presented, along with information on IDS record organization.

DISC STORAGE

IDS implementation for the Compatibles/200/400/600 computers utilizes the DS-15, DS-20, and DS-25 Disc Storage Units. The DS-15 has a removable disc pak. The DS-20 and DS-25 have permanent recording surfaces. However, they are conceptually similar, and a brief discussion of the DS-20 will be sufficient to relate the manner on which the Integrated Data Store stores records on a DSU.

DS-20

Each DS-20 contains a maximum of 16 circular discs. Both sides of each disc are used, giving 32 recording surfaces. Information is recorded as magnetized spots on concentric tracks. Each disc surface is divided into an inner zone and an outer zone. There are 128 circular tracks in each zone, making a total of 256 tracks on each side of each disc. The 128 outer tracks are divided into 16 sectors, and the 128 inner tracks are divided into 8 sectors. Figure 8 shows the format and the manner in which sectors are numbered on disc surfaces.

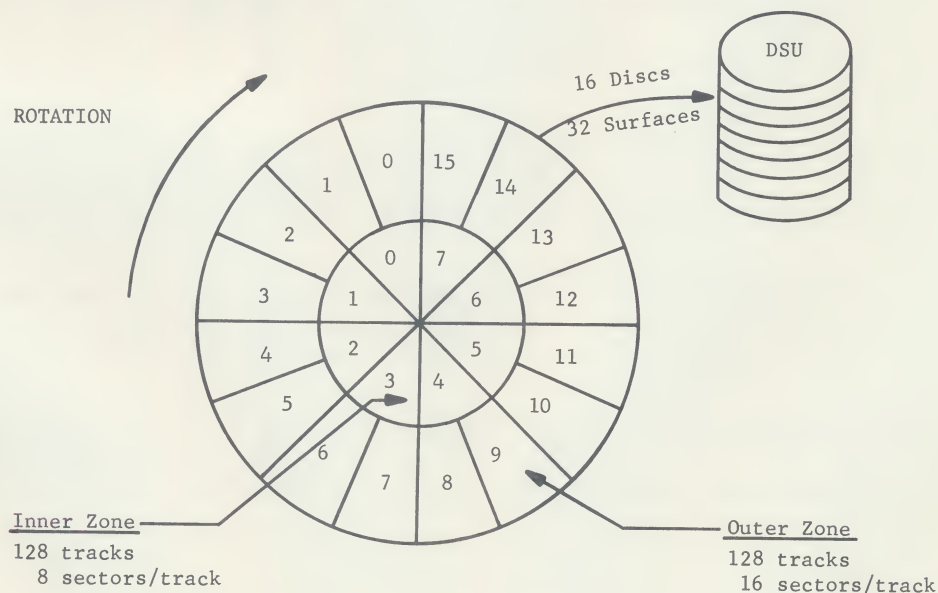


Figure 8. Disc Surface Configuration

Read/Write System

Each disc is served by a separate movable positioning arm. Each arm contains eight read/write heads, four for the upper surface of the disc and four for the lower.

Sectors are continuously addressable within each individual disc file. Up to 32 (16 on the GE-200 Series) contiguous sectors can be read or written with one instruction from the central processor. With the arm in one position, it is possible to transfer a total of 96 sectors.

For each sector on the disc for information storage, there is permanently recorded a sector (record) address. When a search is made to locate the correct sector for a read or write operation, the physical address and the address contained in the instruction are compared. When the two addresses are confirmed electronically as identical, the addressed sector is read or written. The disc address tells where a record is physically located in the file.

Logical records are those records designated by the systems designer/programmer--for example, inventory record or payroll record. They contain data fields pertaining to a specific segment of the application system. Logical records are composed of data and chain fields. The total combined length of a logical record may be any length up to the capacity of a data block (IDS page) which is typically 8-16 sectors.

Conventional disc organizations (not IDS) specify one or more logical records to be contained in a sector, or one or more full or partial sectors to be equal to a logical record. Examples of conventional disc records are shown in Figure 9.

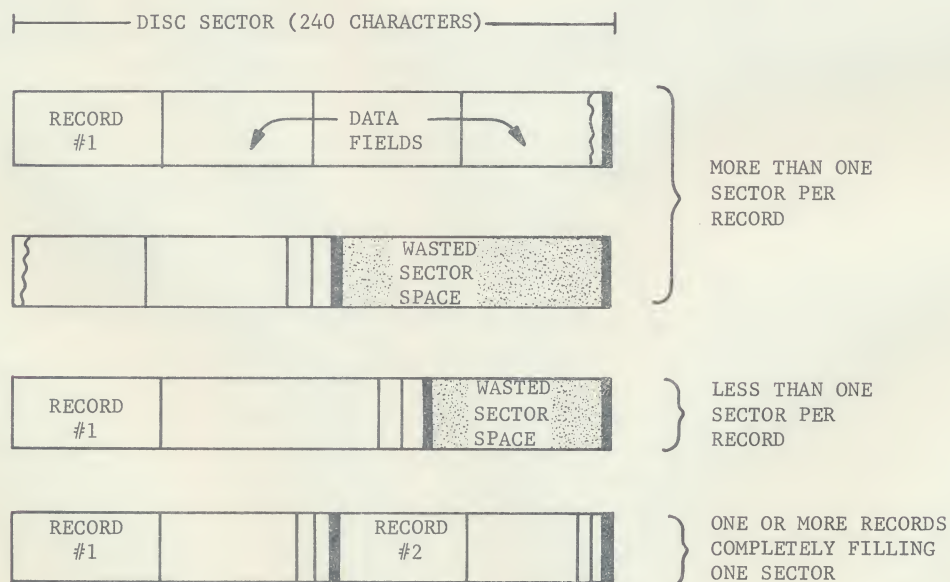


Figure 9. Conventional Disc Records

Note that where equality, or multiples thereof, does not exist (normally the case), inefficiencies in disc storage utilization result.

IDS Pages

In the IDS system a page (or block) of records consists of a fixed number of disc sectors as assigned by the program implementor. A page may contain any combination of logical record types linked into their respective chains. Each type of record has its own specific length. Related record types are associated and linked according to their data content and may be stored within the same page. Space is fully utilized by the automatic packing of these records within the page. During processing, whole pages are read into memory. Therefore, with proper data organization, the probability is high that much of the related information will be available in a single file access. Figure 10 is an example of an IDS page.

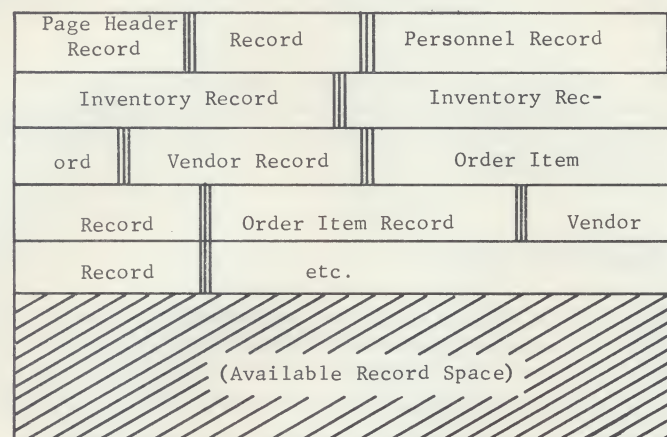


Figure 10. IDS Page

The organization of the "page" facilitates file maintenance and processing, as well as making it possible to accommodate a variable number of record types without duplication of information.

Every page begins with a unique page "header" record. This record contains several control fields used by the system, as follows:

1. Reference address of the page
2. Space available in the page for additional records
3. I/O control indicating whether page has been altered since retrieval
4. Chain field indicating address of the first record of a chain of calculated records, all of which randomized to this page.
5. Line numbers available for assignment within the page

Records within the page contain the following control fields in addition to their user-specified data fields:

1. Line number
2. Record type
3. Record length

REFERENCE CODE

The reference code is a relative addressing code permanently assigned to each record. It can be considered to be a logical address in contrast to an address that defines where the record is physically located in the file. Once a record is assigned a reference code, it maintains that reference code regardless of expansion, contraction, segmentation, or other basic reorganization of the record or file. Thus, the reference code can be used for direct record retrieval on a long-term basis.

The reference code consists of the page number and the specific line number of a data record contained within the page as explained below:

1. The page number portion, a sequential number permanently assigned to each page, which defines what proportion of the way through the IDS page environment the page is stored. Each record stored within a page shares the same page number as part of its reference code. Page numbers are converted to actual disc addresses by the IDS mapping routine at execution time.
2. The line number portion, is a numbering system of records within each page. As records are stored in a page, an available line number (line numbers are reused as records are deleted) is assigned to the record and is combined with the page number to complete the reference code.

INPUT/OUTPUT CONTROLLER

The input/output controller of the IDS controls the mass storage device. Its major function is to control the flow of pages of records in and out of memory in response to commands to store, retrieve, modify and delete specific data records. It also controls the page processing function within memory. Figure 1 illustrates the functions of the input/output controller.

Data Buffers

To minimize the mass storage seek and transfer time, an inventory of data pages is maintained in memory. These pages are stored in numerous buffers in memory. The number of buffers depends on the amount of space available after the IDS subroutines and the problem-solving routine have been loaded.

The greater the number of data pages stored in core memory, the greater the possibility that the one needed next will already be there. To further improve the possibility of finding the page desired in memory, the input/output controller keeps track of the sequence of page utilization (record activity) and holds the most recently active pages in memory. Pages which are infrequently accessed are retired from memory as others are called in. The input/output controller notes which pages have been modified and writes only the modified pages back to mass storage.

Priority Control

Each time a new page is brought into memory, its page number buffer location is placed at the head of a page list. If a page already in memory is used again, it moves to the head of the list. Thus, this list tends to hold the most frequently used pages at the top of the list and the pages with little or no recent use at the bottom of the list. Page space is maintained to allow for the input of a new page. Following input, the page at the bottom of the new page list is automatically written back to the disc storage file (providing there has been activity updating that page).

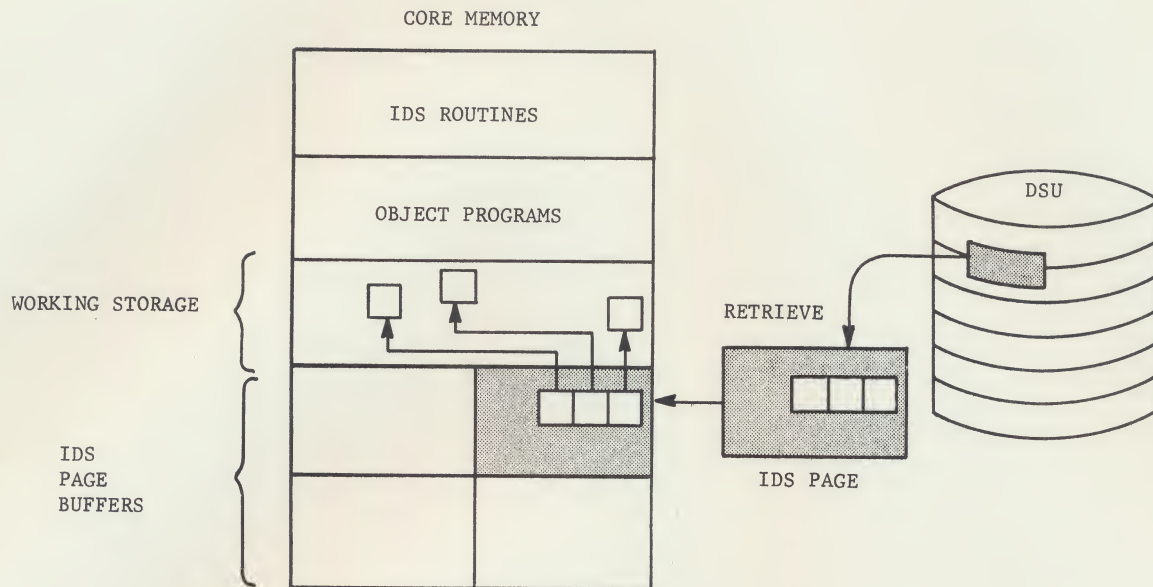


Figure 11. Input/Output Controller

Record Unpacking

Once the input/output controller finds a place for the page in memory, it locates the record called for in the page. The fields from the record will be unpacked into working storage if a **MOVE TO WORKING STORAGE** command is specified.

File Protection

The automatic control provided by the input/output controller for bringing records into memory, writing from memory to disc storages and making the record available for the programmer in the working storage area, eliminates to a large extent the possibility of erroneous updating of record fields. File integrity is therefore maintained through the elimination of record maintenance by the programmer and by restricting the programmer to the updating of record fields in the working storage area only. The modify function, which can address data fields, provides the only way of changing an actual record.

4. ELEMENTS OF IDS

This chapter describes the basic functional elements of IDS. These elements are best described if divided into two areas of discussion:

1. Data organization
2. Input/output controller

DATA ORGANIZATION

Data organization refers to the interrecord relationships established within the IDS. The record is the basic unit of data. Record association is achieved through chains which provide cross-reference linkages between records. These chains provide the integrating force which is implied in the name Integrated Data Store.

As shown below, the IDS record contains a set of "data fields" which collectively describe the event, activity, status, or plan that the record represents. IDS augments these records with additional fields called "identification" and "chain" fields, as shown in Figure 12.

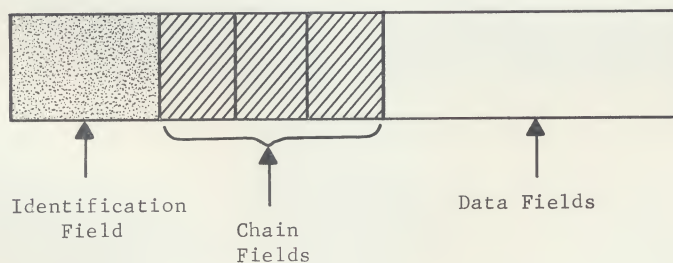


Figure 12. IDS Records

The chain fields contain the reference codes of other IDS records. They point from one record to the next, creating a circular association of records (Figure 13).

These associations are automatically processed according to the data descriptions and the procedural commands executed. The arrows in Figure 13 indicate the linking actually carried out through the storing of the reference code of one record in the body of the prior record.

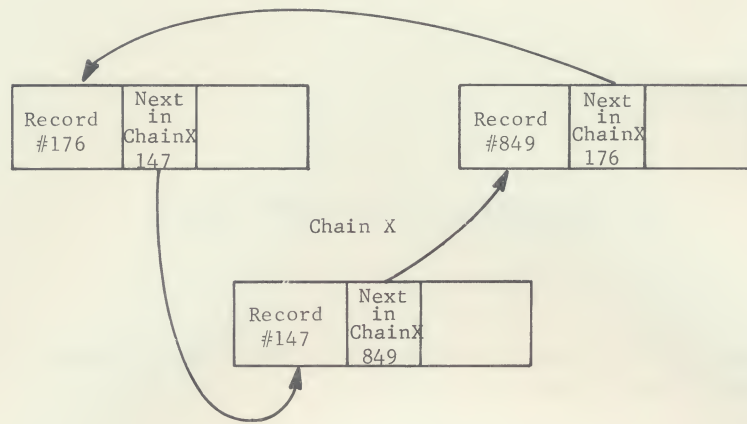


Figure 13. Chain Association

Data Records and Fields

The records of the IDS are fixed-format, fixed-length records in the COBOL tradition; that is, the length and format of a specific type of record, such as payroll or inventory record, are fixed by the specifications of the systems designer. Records of many different types, each with its own length and format may be used in the system. In order that control may be maintained, each record has the same identification fields at the very beginning. These fields are (1) line number position of the reference code, (2) record type (such as inventory, payroll, etc.), and (3) record length. The rest of the record consists of data and chain fields to suit the application requirements.

Records may have any number of data fields, each defined as some number of decimal, alphabetic, or alphanumeric characters. Fields may vary in size from one character up to many characters, as for a drawing or part number or an employee's name. These fields will be specified by the systems designer.

Chain fields contain the reference code for addressing and are defined for each chain in which a record participates. Experience in IDS systems indicates that the average record is in only two chains. An occasional pivotal record in the information integration may be in six or eight chains. There is no upper limit on the number of data or chain fields except that provided by the maximum page size. Average record size has been 30-40 characters in total length, with an occasional record of 120-180 characters.

IDS records are stored only once in IDS. Conventional approaches to file organization often require records, or certain fields in the records, to be repeated in several files. With the ability to link records into any number of chains (as required by the system), the same data fields are available for all chains processed. This technique of eliminating redundant data has four important advantages:

1. Additional space required for duplicate records is eliminated, resulting in a reduction in the total capacity required.
2. The work of data maintenance is greatly reduced, as there is only one record to retrieve and modify.

3. The possibility that one of the copies of a record will not be properly modified is eliminated. Since there is only one copy of a record, any incorrect information will be quickly recognized and corrected.
4. All reports drawn from the file will be consistent, since there is only one set of facts (records).

Record Classes

The Integrated Data Store recognizes three distinct classes of records that it must store and retrieve. The designation of the data records as to class is at the option of the systems designer and is based on the storage and retrieval requirements of these data records.

IDS record processing requires that there be some aspect of every record which makes it unique, or different from any other record. All records are unique by virtue of their reference code. Some records are also unique because they contain one or more data fields--such as a drawing number, order number, and pay number--where no duplicate values are allowed.

Calculated Records. Any record within the system can be classified as a "calculated" record. Its storage and retrieval are based upon the contents of one or more data fields. The contents of these fields are externally specified numbers--such as employee numbers, part numbers, or order numbers. The contents of these fields are processed through a randomizing procedure which determines a page number for an initial storage location. The record is stored at this page or one very close to it. A purchase order or an inventory item record is typically defined as this type of record. The subsequent retrieval of this record follows this same basic procedure.

Secondary Records. "Secondary" records are a class made unique by virtue of their chain relationship to a specified type of master record and a sort control field(s) used to sequence that chain. The item records associated with a purchase order (master) record are a good example of secondary records (refer to Figure 7). These records are stored and retrieved by first locating the purchase order record and then stepping through the order item chain to locate the item record by comparison of its item number field.

Primary Records. Records designated in the data description as "primary" are unique only as a result of their reference codes. Generally "primary" records are used in place of "calculated" records where the external assignment of identification fields, such as, part number or order number is not required. In these cases, an internally generated number (the reference code) is assigned and used as the identification field

IDS Chains

An IDS chain is illustrated in Figure 14. Its characteristics are:

- Has one master record and any number of detail records
- Links records together in an endless loop
- Associates related records in meaningful sequences.

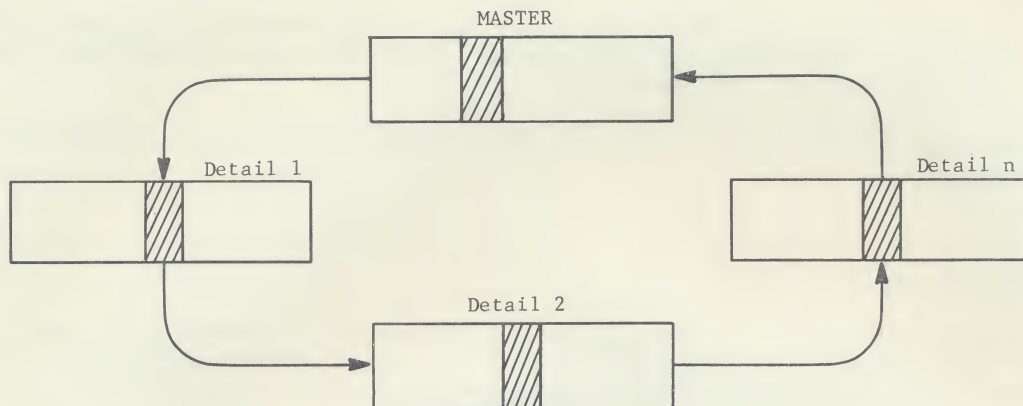


Figure 14. IDS Chain

In addition to records being designated as to class for storage and retrieval purposes, they must also be defined as to their relationship within a chain--master or detail records. These record relationships are specified, when the chain is defined, in the data description.

This master-detail relationship of records is analogous to the conventional "header-trailer" file sequence of records. The master record of the chain describes the fixed information (header type) pertaining to the variable number of detail records in the chain. The detail records in the chain describe the variable information pertaining to the master record.

Multiple Chains

Chains exist for two separate but closely related reasons. First is the case where the source documentation shows that a portion of the information appears in multiples--for example, a personnel record with a variable number of deductions and work experiences.

This type of information is easily structured by building a personnel master record. Two chains are created containing the personnel record as the master record. As many deduction records as necessary are linked into the deduction chain as details. Work experience for the employee involved is handled in a like manner. Both chains are now linked to the same master record as shown in Figure 15.

The second case for chain structuring of information involves the association of several related source documents. Relating all the purchase orders for a given vendor is an example. Figure 7 (page 9) illustrates this example. The purchase order chain associates all of the purchase order records with their vendor record.

IDS chains have several structural aspects which should be emphasized:

A chain type is named with a symbolic name. There will be as many chains of the chain type as there are records of the type which serve as the master of the chain type.

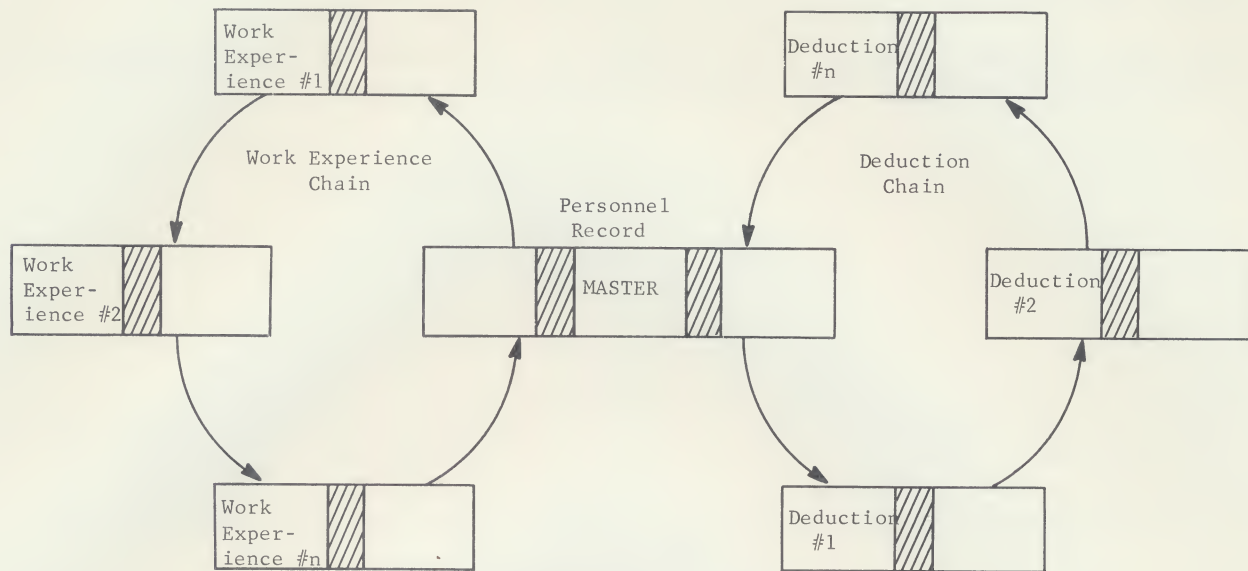


Figure 15. Master Record of Two Chains

Each chain has only one master record. The record type of the master record is the same for all chains of the same type.

A chain is created whenever its master record is stored in the IDS. The chain is destroyed when the master record is deleted.

A chain may contain more than one type of detail record. Any number of detail records may be in a chain.

Detail records cannot be stored unless there exists a master record which qualifies as the particular master according to the specified master selection rule for that chain.

Whenever a master record is deleted, all of its detail records are automatically deleted (and their detail records too).

All records in a chain are associated in an endless loop with the last detail chained back to the master.

The master record of the chain stores the reference code of the first detail in the chain.

A record (detail or master) may be defined to be in as many chains as are required. It may be defined as master in one chain and detail in another.

A record cannot be defined as a detail to itself directly or indirectly.

As records are stored in the system, they are automatically linked into their defined chains.

When a record is deleted, the chains in which it is a detail record are automatically patched to be relinked around the deleted record.

Chain Processing

IDS offers complete flexibility in the retrieval of records within a chain by providing three methods of chain processing. These methods are illustrated in Figure 16.

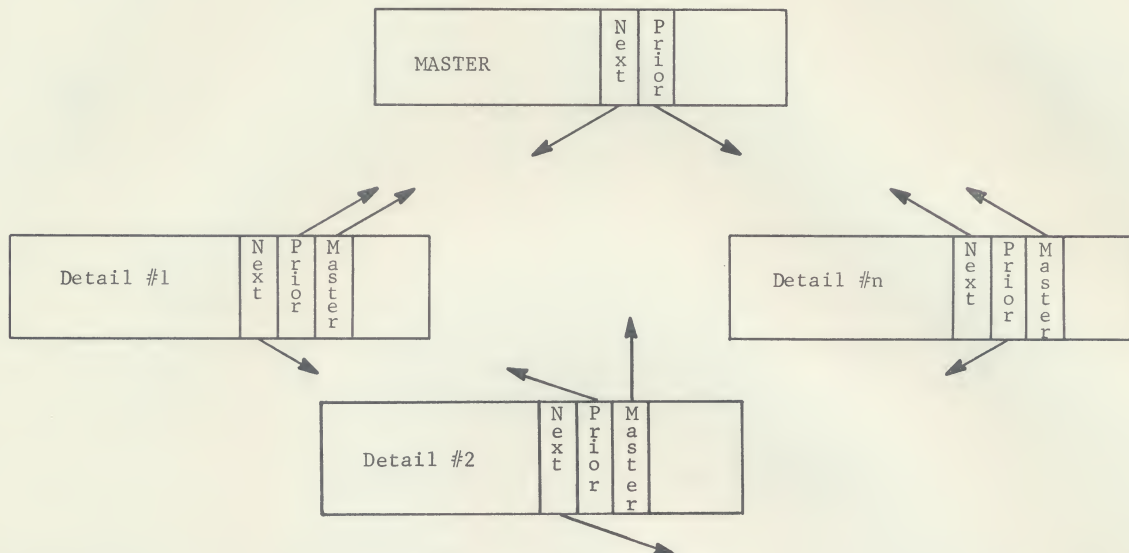


Figure 16. Chain Processing

1. Chain Next--The definition of a record in a chain automatically provides the record with a chain-next field. This is the manner in which all chains are constructed. Each record contains a chain-next field which contains the reference code of the next record in the chain.
2. Chain Prior (optional)--IDS provides a chain-prior field for all records in a chain when the chain is specified by the systems designer as prior processable. This field contains the reference code of the prior record in the chain. This permits the chain to be processed efficiently in a backward direction.
3. Chain Master (optional)--IDS provides a chain-master field for all detail records in a chain when specified in the data description. This field contains the reference code of the master record of the chain. Retrieval of the master record is much faster with this ability to address directly the master record from any detail in the chain. Processing need not access all the detail records in the process of seeking the master.

Linking a New Detail Record

In order to insert a new detail record in a chain, two steps are required:

- The appropriate master and its chain must be selected.
- The record must be inserted in that chain according to the chain ordering rules.

Master Record Selection

There are two rules under which the master record is selected for a new detail record. These are:

- Select Unique Master--This rule uses the record retrieval criteria, established in the data definition for the master record, to retrieve the particular master record indicated by the data values currently stored in the match control field of working storage.
- Select Current Master--This rule uses the last record processed, of the master record type, as the master record of the new detail.

Chain Ordering

All chains in the Integrated Data Store system are ordered in one of six methods specified by the systems designer with the chain-order clause in the IDS language.

The chain-order clause must be used in each detail chain definition entry except, when a record is defined as a detail of the CALC chain.

The six options of the chain-order clause are:

1. Sorted Within Type--With this option the records of the chain are maintained in sequence within record type, independent of other types.
2. Sorted--With this option the various records of the chain are maintained in a single sequence regardless of the number of record types in the chain. With this option the control fields of the various records must be of identical size.
NOTE: When either of the sorted options are specified, details are added to the chain based upon the content of the defined sort control fields of the detail records.
3. First--This option causes the detail to be added as the first detail record in the chain relative to the master record.
4. Last--This option causes the detail to be added as the last detail record in the chain relative to the master record.
5. Before--This option causes the insertion of the detail record just before the current record in the chain. This option may only be used in conjunction with the "Current Master" selection rule.
6. After--This option causes the insertion of the detail record just after the current record of the chain. This option may be only used in conjunction with the "Current Master" selection rule.

Prime Chain

Access time in present disc-type random access memories varies greatly, since it depends on the position of the desired record relative to the record last accessed. The IDS organization of records acknowledges this factor of hardware design and stores new detail records as close as possible to the master record of the chain. When a detail record is specified as a detail in several chains, a prime chain may be chosen and defined by the systems designer preparing the data

description. Selection of a prime chain should be based upon an estimate of the most active chain. Thereafter when an IDS page is retrieved which contains the master record of a prime chain, it is highly probable that the detail records of that chain will also be contained in that page or a page closely associated with it. The prime chain is the chain used to retrieve a secondary record by the RETRIEVE command, unless specified otherwise by the data description.

Data Structure

A special graphic technique called "IDS shorthand" has been developed to display records and their master-detail (chain) relationships. Its use is particularly important in developing an overall view when planning an information system. This technique (see Figure 17) uses a block shape to designate a record type--employee (record type 1) and deduction (record type 2)--and an arrow connecting two blocks to designate a chain. The arrow points from the master to the detail, as shown in Figure 17.

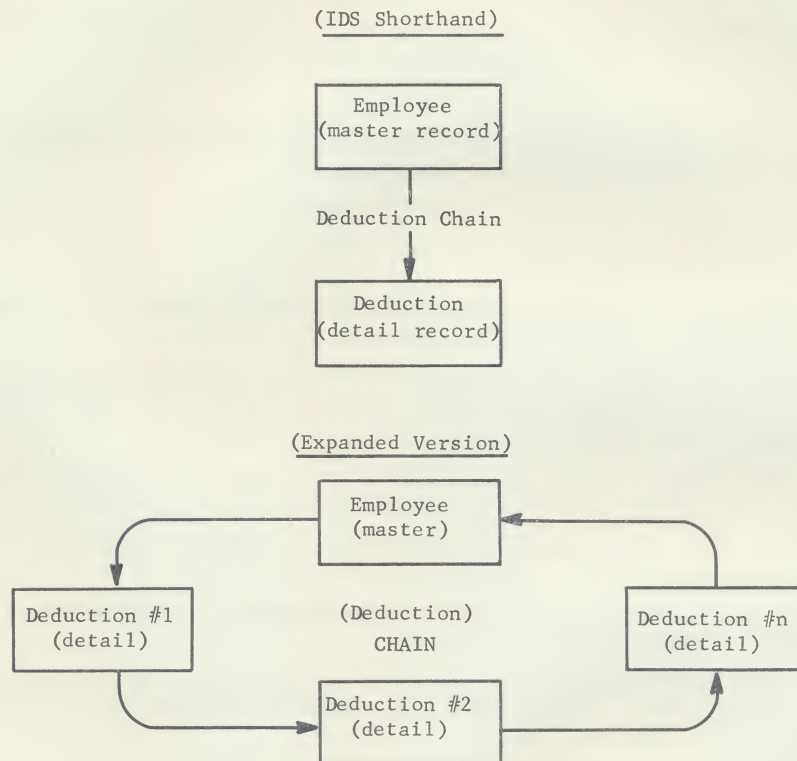


Figure 17. IDS Shorthand

In the foregoing example of IDS shorthand, the vertical block-arrow-block sequence carries the following message:

1. There are a number of records in the system of the master type (one for each employee).
2. Each of these records is the master of a chain of the specified type (deduction).
3. There are a number of records of the detail type (deduction 1, 2, 3, 4, etc.) in each such chain.

We will now take a vendor record and a particular order record from the initial example (Figure 7) and show how this information is normally structured in the IDS system. This is shown by the purchase order data structure (Figure 18).

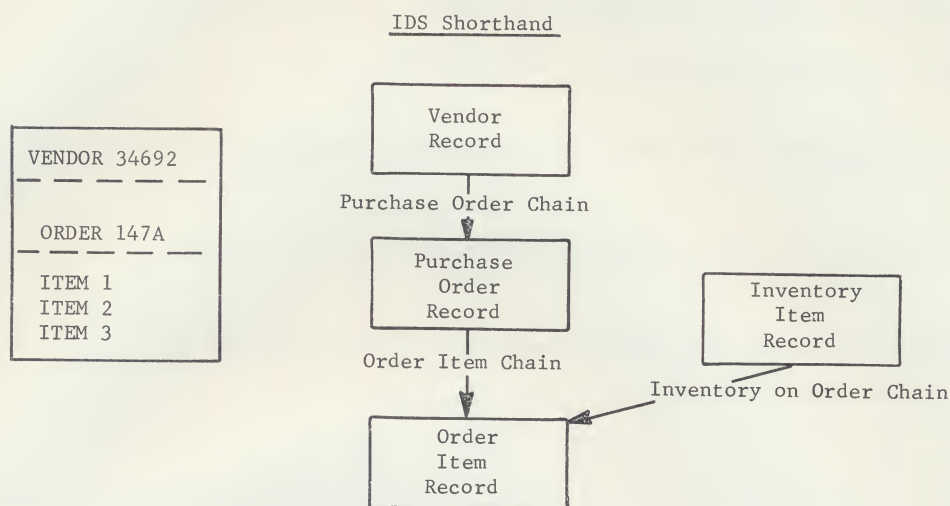


Figure 18. Purchase Order Data Structure

In looking at the purchase order four groups of information can be seen. These are:

1. Information about the vendor--such as his name, address, and vendor code.
2. Information about the order--such as the order number, due date, mode of transportation, and dollar value.
3. Information about the order item--such as delivery date, quantity, unit price, and extended dollar value.
4. Information about the inventory item--such as its identification and description.

The data structure in Figure 18 shows all four groups and their chain associations with only four blocks and three arrows. To expand this structure, four different record types would be designed to carry the information contained in the four groups:

Vendor record--There would be a vendor record for every vendor with whom the business is concerned:

1. It would be the master record of a purchase order chain
2. Thus, the vendor record is only a master.

Purchase Order record--There would be an order record for each order currently stored in the system:

1. It would be a detail in the purchase order chain
2. Each order, in turn, would be the master of an order item chain
3. Thus, the purchase order record is both a master and a detail.

Order Item record--There would be an order item record for each item on each order.

1. It would be a detail in the Inventory on Order chain.
2. It would be a detail in the Order Item chain.
3. Thus, the Order Item record is a detail in two chains.

Inventory Item record--There would be an inventory record for each Inventory Item currently stored in the system.

1. It would be the master record of the Inventory on Order chain.

The expanded data structure for the above records is shown in Figure 19.

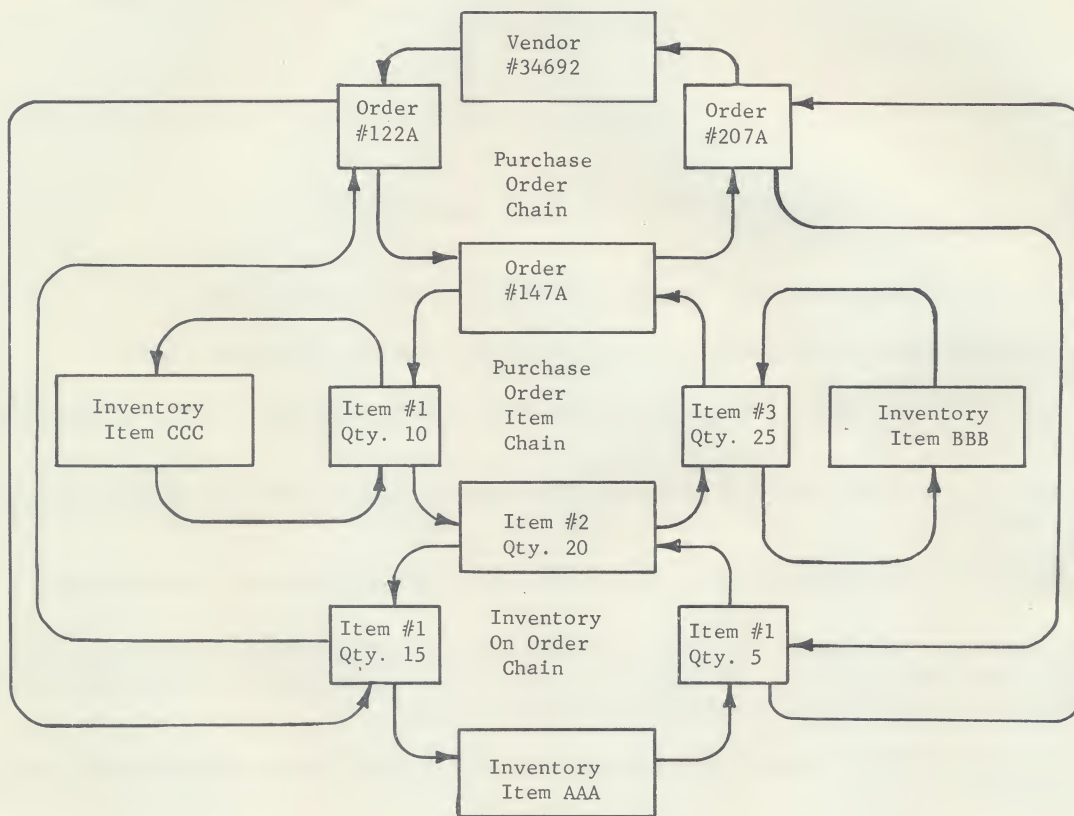


Figure 19. Chain Network

Summary of Data Structures

By using IDS shorthand, very complex data structures may be presented in a condensed and understandable form. Thus, the following examples (Figures 20 and 21) show a quick summary of data structures which are legal within IDS.

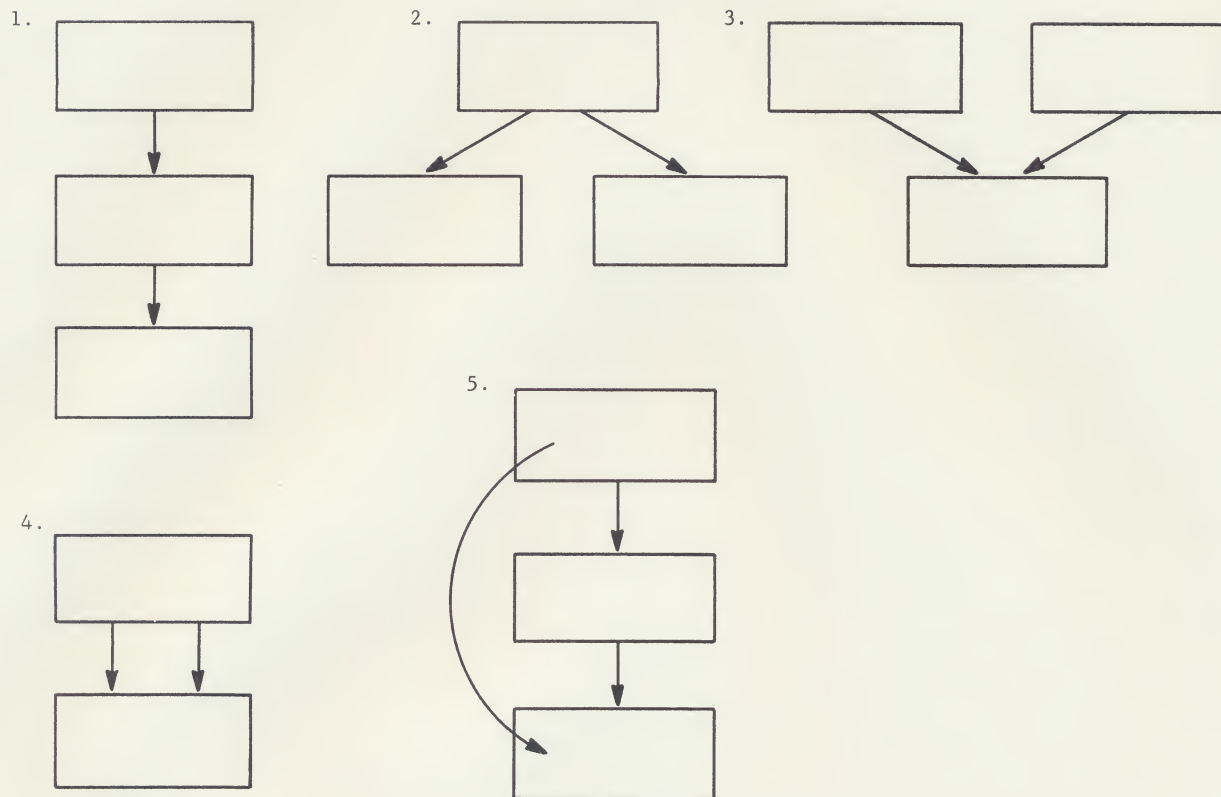


Figure 20. Legal IDS Structures

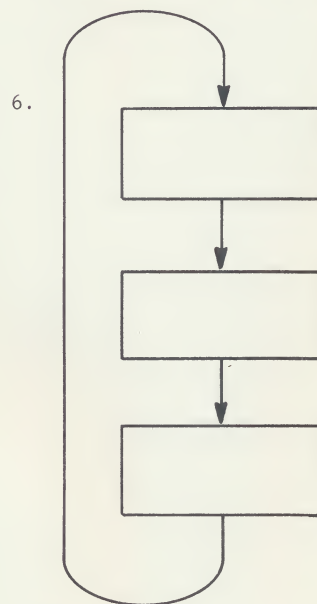


Figure 21. Illegal IDS Structure

NOTE: A circular definition, as shown in Figure 21, is not allowed.

5. RECORD PROCESSING LANGUAGE

IDS provides its user with the ability and requirement to predefine his records, their data fields, and their chain fields. Once these records and fields have been defined, the user is free to operate upon the records without concern for the physical aspects of input or output, the linking of records into chains, or the protection of the data from erroneous access.

All hardware manipulation and information processing control of the disc storage unit is carried out automatically to achieve the four major processing functions:

STORE--Stores information and links it into chains
RETRIEVE--Retrieves information from the file
MODIFY--Changes or updates information, and relinks chains where necessary
DELETE--Removes information from the file and relinks chains.

SOURCE LANGUAGE

The source language of IDS is implemented as an extension of COBOL on the GE-400/600 product lines. The GE-200 version is implemented through the use of calling sequences and subroutines within the General Assembly Program. All formats and language specifications of COBOL must be adhered to when preparing a source program.

The purpose and usage of the IDENTIFICATION DIVISION is just as defined for COBOL with no special function so far as IDS is concerned.

All portions of the ENVIRONMENT DIVISION with the exception of the FILE-CONTROL paragraph of the input/output section are used as defined by COBOL.

To define the file-name which represents the IDS data file, a unique version of the SELECT sentence is required as shown below:

FILE-CONTROL. SELECT IDS file-name

The format of the IDS data description is an extension of the COBOL Data Division. As in COBOL each record and data field within each record is described. In addition, certain parameters required to guide the storage and retrieval process are supplied for each record. These include a definition of the various chains in which each record may be either a master or a detail.

COMPATIBLES

IDS

The procedural language of IDS consists of the two basic verbs, STORE and RETRIEVE. These two verbs together with their several variations are expressed in terms similar to those used by COBOL.

The STORE verb stores a new record in accordance with its data description. The data content of the record is moved from working storage and placed into a page. The chain fields are initialized to reflect the record's position in each of the chains defined for the record. The record becomes the current record of its type and also the current record in each chain in which it is defined.

The RETRIEVE verb may take one of several different forms dependent upon the record to be retrieved. The alternatives are:

1. RETRIEVE record-name. In this case retrieval is accomplished in accordance with the data description for the record named. This is called associative retrieval.
2. RETRIEVE CURRENT record-name. In this case the record retrieved is the last one processed of the type named.
3. RETRIEVE $\left\{ \begin{array}{l} \text{MASTER} \\ \text{NEXT} \\ \text{PRIOR} \end{array} \right\}$ of chain-name

In this case the record retrieved is a relative to the "current" record's in the chain named.

4. RETRIEVE DIRECT. This case may be used when the reference code of a record is known and has been stored in a communication cell.
5. RETRIEVE EACH AT END GO TO procedure-name-1. In this case a sequential search of the storage device will retrieve the first record in the file within a range of reference codes.

In each of the above cases the record to be retrieved will be made available in memory, subject to additional functions which may be appended to the basic RETRIEVE statement. These additional IDS functions include the MODIFY and DELETE statements.

The additional IDS functions are listed below:

1. MOVE the record to working storage.
2. MODIFY the record by field replacement, by incrementing a field, or by decrementing a field.
3. DELETE a record together with its associated detail records.
4. HEAD chain-name moves to working storage the record which is the master of the current record in the named chain.
5. GO TO or PERFORM some alternate procedure.

It is the responsibility of the associated COBOL procedure to initialize working storage appropriately prior to the use of one of the IDS verbs. When a record is to be stored the complete data of that record must be in its associated working storage areas. When a record is to be retrieved associatively the control fields which uniquely identify that record must have been initialized in working storage.

OPEN

The OPEN verb is used to initialize the data buffer prior to the execution of any other IDS verb.

OPEN IDS file-name.

NOTE:

- File-name must be identical to the one used in the SELECT IDS sentence of the FILE-CONTROL paragraph of the ENVIRONMENT DIVISION.

CLOSE

The CLOSE verb is used to force the writing to the disc storage unit of any pages in memory which have been modified.

CLOSE IDS file-name.

NOTE:

- This statement must be executed before any COBOL STOP RUN statement. No automatic closing will take place.
- File-name must be identical to the one used in the SELECT IDS sentence of the FILE-CONTROL paragraph of the ENVIRONMENT DIVISION.

Procedural Statements

The basic function of the IDS procedural statements is to accomplish the storage and retrieval of data with respect to the mass storage device. In addition these statements have the implicit responsibility to maintain the structure of the data file that is created by the defined chain relationships. The mechanism which accomplishes these functions is the IDS data controller.

The communication interface between the data controller and the balance of the COBOL PROCEDURE DIVISION is the working storage area which is established for each 02 level data field defined by the record description of the IDS section. All COBOL references to data from the IDS file will be to these working storage areas.

The format considerations for IDS statements are the same as those specified for COBOL. The one exception is that an IDS sentence must appear on unique lines of the coding form. Normal rules for continuation apply.

If it is desired to assign a procedure name to an IDS sentence it must appear on a separate line just preceding the IDS sentence. The IDS sentence may consist of as many statements as are required with a period following the last statement.

The structure of an IDS sentence will always be one of the following forms:

1. $\left\{ \begin{array}{l} \underline{\text{STORE}} \\ \underline{\text{RETRIEVE}} \end{array} \right\}$ record specifier.

COMPATIBLES

IDS

2. $\left\{ \begin{array}{l} \underline{\text{STORE}} \\ \underline{\text{RETRIEVE}} \end{array} \right\}$ record specifier; statement-1
- $\left[\begin{array}{l} \text{statement-2. . .} \end{array} \right]$

where statement-1, statement-2, etc. may be either imperative statements or conditional statements.

Note that the STORE or RETRIEVE verb must be the first statement of each IDS procedural sentence. The record specifier may be used only as defined for the STORE or RETRIEVE verbs.

A description of how these various sentence formats may be used and the limitations which apply to each follows.

STORE

FUNCTION: To place a record into the IDS data file and to establish any chain fields which may be required.

STORE data-name RECORD

NOTES:

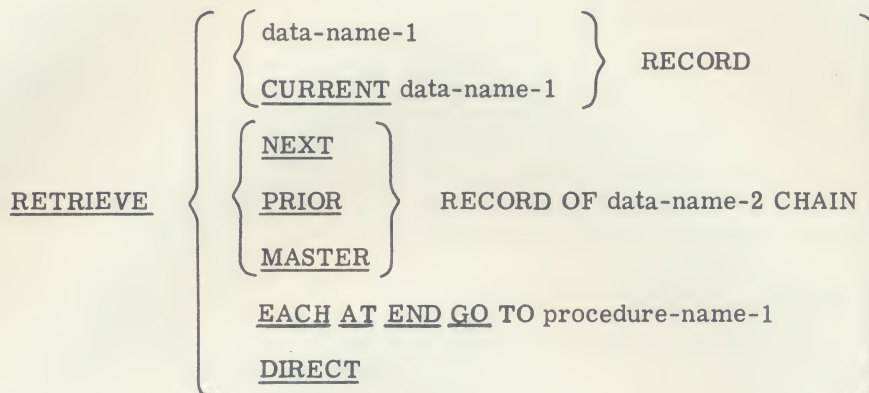
- Data-name must be one defined for a record level (01) entry of the IDS SECTION of the DATA DIVISION.
- Use of this verb assumes that the working storage area for each field of this record has been initialized with the desired field value. It is also assumed that any other control fields required to provide unique identification of the master records of specified chains have been initialized in their respective working storage areas.
- The record is placed into the file as defined by the PLACE or RETRIEVAL clauses of the Record Description entry.
- The reference code assigned to the record stored is left in the communication cell DIRECT-REFERENCE after the storage process is complete.
- The record stored is recorded as the CURRENT record of its type and the CURRENT record in each chain in which it is a master or detail.
- If the storage process creates a duplicate record in violation to any DUPLICATES NOT ALLOWED clause, or if the unique or range master selected, cannot be retrieved, the storage process is terminated with all linkages restored as before and an error condition is noted.

COMPATIBLES

IDS

RETRIEVE

FUNCTION: To cause a record to be made available in the memory buffers.



NOTES:

- The various options of the RETRIEVE verb are hereinafter referred to as the record-specifiers.
- Data-name-1 must be the name of a record (01) level entry defined in the IDS SECTION of the DATA DIVISION.
- Data-name-2 must be the name of a chain as defined by a level 98 entry of the IDS SECTION of the DATA DIVISION.
- Regardless of the record specifier used the action which results from this verb is to retrieve the record referenced and to make it available in the memory buffer. This action may or may not require that a page be transmitted from the mass storage device since the record may already be in memory. No other action such as moving the record to working storage is implied by this verb.
- Of the seven record specifiers which may be used with the RETRIEVE verb, two may be classified as absolute, there is only one record that will satisfy the retrieval specification regardless of when they are executed.
 1. RETRIEVE data-name-1 RECORD. The record retrieved depends on the RETRIEVAL clause defined as a part of the level 01 entry in the IDS SECTION and upon the value contained in the control fields of working storage which uniquely identify the record.
 2. RETRIEVE DIRECT. The record to be retrieved is the one which contains the reference code stored in a communication cell named DIRECT REFERENCE. It is the user's responsibility to initialize the communication cell prior to the execution of this command.

The other five record specifiers may be classified as relative in nature in that the actual record retrieved is a function of what has transpired previously.

COMPATIBLES

IDS

3. $\left\{ \begin{array}{c} \underline{\text{NEXT}} \\ \underline{\text{PRIOR}} \\ \underline{\text{MASTER}} \end{array} \right\} \quad \text{RECORD OF data-name-2 CHAIN.}$
- RETRIEVE

In this case the record retrieved is dependent upon the CURRENT record within the chain specified. In the case of NEXT or PRIOR the appropriate record is retrieved regardless of the record type. When MASTER is specified all intervening records are ignored and the master record of the chain named is retrieved. These record specifiers can only be used if some record has already been processed which is a member of the specified type of chain.

4. RETRIEVE CURRENT data-name-1 RECORD. This record specifier instructs the system to retrieve the last record of the type specified, that was processed by any STORE or RETRIEVE verb. If the last record processed had been deleted, it would not be available and an error condition would exist.
5. RETRIEVE EACH. This record specifier provides for a serial search of the mass storage device beginning with the reference code contained in a communication cell called FIRST-REFERENCE. This command retrieves the first record found unless the reference code of the record retrieved is greater than the contents of a communication cell called LAST-REFERENCE. At this point the system exists according to the AT END conditional statement. It is the user's responsibility to initialize the communication cells with the appropriate reference codes. The user may retrieve all of the records in the file through the repeated use of this record specifier. As a record is retrieved the reference code value contained in the FIRST-REFERENCE cell is incremented by one so that subsequent use of this verb is initialized.

- The reference code of the record retrieved is placed in the communication cell named DIRECT-REFERENCE after the retrieval process is complete.
- The record retrieved is recorded as the CURRENT record of its type and the CURRENT record in each chain in which it is a master or detail.
- If a record cannot be retrieved according to the specifications of the retrieval command, an error condition will be noted.

Imperative Statements

The imperative statements included in this section are provided as a part of the IDS language to extend the function of the basic STORE and RETRIEVE verbs. The MOVE, HEAD, MODIFY and DELETE statements apply only to the RETRIEVE verb whereas the GO and PERFORM statements may be used with either verb.

These statements will be executed in the sequence in which they occur. They must be contained within the sentence beginning with the basic verb and ending with a period. They may not be used as separate sentences to accomplish their intended result.

The specific formats of these statements together with a detailed discussion of their associated restrictions and limitations follows.

MOVE

FUNCTION: To cause the record retrieved to be moved from the buffer to working storage.

; MOVE TO WORKING-STORAGE

NOTES:

- The implied source of the MOVE is the record just retrieved and available within the buffer.
- This statement must be used before any reference can be made to the data in the record.
- When the record is moved to working storage, each data field is unpacked into an integral number of words as if the fields had been defined as SYNCHRONIZED. These fields are SYNCHRONIZED-RIGHT unless SYNCHRONIZED-LEFT was specified in the data description.

HEAD

FUNCTION: To enable the automatic retrieval and move to working storage of any higher level master record in the hierarchy which includes the record just retrieved.

; HEAD data-name CHAIN [; HEAD. . .]

NOTES:

- Data-name must be the name of a chain as defined by a level 98 entry of the DATA DIVISION. Further, the chain must include the record just retrieved.
- This statement includes an implied move of the record retrieved to working storage.
- After execution of this statement the master records retrieved are the CURRENT records of their respective types. They become the CURRENT record in each chain in which they are defined as details. However, they will not be the CURRENT record in the chains in which they are defined as master records. In these chains the detail record which leads to them is the CURRENT record.

MODIFY

FUNCTION: To modify the contents of a field of the record retrieved and to maintain any chains which may be controlled by the field modified.

; { ADD
SUBTRACT
REPLACE } data-name FIELD [; { ADD
SUBTRACT
REPLACE } . . .]

COMPATIBLES

IDS

NOTES:

- The field to be modified must be defined as a 02 level field in the IDS SECTION of the DATA DIVISION.
- This statement causes the contents of working storage to be added to, subtracted from, or to replace the contents of the field specified in the record in the buffer. The results of the change also appear in working storage.
- Data-name may be any field contained in the record specified by the RETRIEVE verb. In addition, data-name may be a field that has been defined as a MATCH-KEY field by a detail chain definition (level 98) entry associated with the record retrieved. The only exception is the data-name used for primary retrieval--an attempt to modify this field will result in an error.
- When the field to be modified is a MATCH-KEY field the result is to change the association of the detail record with its master. The record is delinked from its old master, its new master retrieved, and the record linked to its new master according to the CHAIN-ORDER clause for the chain. Note that in this case there is no actual change of the field specified, but instead the record has become a part of a different chain.
- It is also possible to change a field in the record retrieved if that record is defined as a sequence key. In this case the field is changed, the record delinked from its master and relinked to it again in accordance with the new value of its sequence key field.
- It should be noted that when the field to be modified is a control field such as the cases mentioned in notes 3 and 4, a conflict in the use of working storage can occur. When the retrieval action is accomplished by the RETRIEVE data-name option, the appropriate working storage fields for the various control fields had to be initialized with the values currently contained in the data record. Since the modify function assumes the change values are in working storage a conflict is apparent. To resolve this problem the user should: RETRIEVE data-name RECORD; initialize working storage with change values; and then RETRIEVE CURRENT, REPLACE field-name. Or, the user may: RETRIEVE data-name RECORD; PERFORM procedure name then modify after the procedure named has changed working storage.
- When the ADD or SUBTRACT option is used, the field named must have been defined as a numerical field in the IDS SECTION of the DATA DIVISION.

DELETE

FUNCTION: To delete the record retrieved together with all of its dependent detail records and to optionally perform certain functions on the occurrence of specified detail record types during the deletion process.

```
      ; DELETE [ ON data-name-1 DETAIL
      [ MOVE TO WORKING-STORAGE ]
      [ HEAD data-name-2 CHAIN [ HEAD . . . ] ]
      [ PERFORM procedure-name ]
      [ GO TO procedure-name ] ]
      [ ( OTHERWISE )
      [ . ELSE ] ] ON data-name-3 DETAIL . . . ] .
```

NOTES:

1. The implicit object of the DELETE statement is the record retrieved by the RETRIEVE verb.

COMPATIBLES

IDS

2. The deletion process only deletes a record when there are no dependent details in its chains. When details are present, the system first attempts to delete the dependent detail records. Since the hierarchical data structure of IDS may involve many levels of detail records, this statement assumes powerful capabilities and should be used with due consideration.
3. The execution of a DELETE statement makes the record retrieved unavailable for any further processing and an attempt to reference such a record results in an error condition.
4. The conditional statement ON data-name-1 DETAIL is used only when it is desired to interrupt the deletion process when a dependent detail of the type named by data-name-1 is encountered. When this is the case, the various imperative statements immediately following are executed prior to the actual deletion of the detail record. After the execution of these statements, the deletion process is continued unless one of the statements is a GO TO statement. In that case, control is not returned to the deletion process. When the record encountered is not the type named by data-name-1 it will be compared with the type named by data-name-3. The reserved words OTHERWISE or ELSE serve to separate the tests for different record types that may be encountered. A record encountered which does not match any of the specified record types will be deleted in the normal manner.
5. As a record is deleted it is not implicitly moved to working storage.

GO

FUNCTION: To depart from the normal in-line sequence of procedures.

; GO TO procedure-name-1

NOTES:

1. Procedure-name-1 may be any COBOL or IDS procedural paragraph in the PROCEDURE DIVISION.
2. When this statement is encountered within the IDS sentence, all subsequent statements are bypassed and control transferred to the procedure named.

PERFORM

FUNCTION: To depart from the normal in-line sequence of procedures in order to execute a specific procedure and then return to the normal sequence.

; PERFORM procedure-name-1

NOTES:

1. Procedure-name-1 may be any COBOL procedural paragraph in the PROCEDURE DIVISION. However, only the simple PERFORM (option-1) is recognized within an IDS sentence.
2. No procedure may be specified which directly, or indirectly calls upon another IDS command.

COMPATIBLES

IDS

Conditional Statements

The conditional statements of IDS are a logical extension of the basic STORE and RETRIEVE verbs. Generally they involve the key word IF followed by the condition to be tested followed by the imperative statements to be performed.

IDS conditional statements are of two general forms, and may appear in the string of statements following a basic verb only as the last statement of the string.

The specific formats of these statements and a discussion of their restrictions and limitations are outlined below and on the following pages:

Form 1.

$$\begin{aligned} & ; \text{ IF data-name-1 } \underline{\text{RECORD}} \left\{ \begin{array}{c} \underline{\text{NEXT SENTENCE}} \\ \text{statement-1 [; statement-2. . .]} \end{array} \right\} \\ & \left[\left\{ \begin{array}{c} \underline{\text{OTHERWISE}} \\ \underline{\text{ELSE}} \end{array} \right\} \left\{ \begin{array}{c} \underline{\text{NEXT SENTENCE}} \\ \text{statement-3 [; statement-4. . .]} \end{array} \right\} \right] . \end{aligned}$$

NOTES:

1. This form may only follow a RETRIEVE DIRECT, RETRIEVE EACH or RETRIEVE $\left\{ \begin{array}{c} \underline{\text{NEXT}} \\ \underline{\text{PRIOR}} \end{array} \right\}$ RECORD OF data-name CHAIN.
2. Statement-1, statement-2, etc., may be any imperative statements. Statement-3, statement-4, etc., may be any imperative statements, or a conditional of this form (form 1).
3. This form of conditional is an implicit comparison of the record type just retrieved, with the record type named by data-name-1. If the record type currently available equals the record type named, the condition is true and statement-1, statement-2, etc., are executed in order. Then control is transferred to the next sentence. If the condition is false statement-3, statement-4, etc., are executed. If the clause beginning with the key words OTHERWISE or ELSE is omitted and the condition is false, control is transferred to the next sentence.

Form 2.

$$; \text{ IF } \underline{\text{ERROR}} \text{ statement-1 } \left[\left\{ \begin{array}{c} \underline{\text{OTHERWISE}} \\ \underline{\text{ELSE}} \end{array} \right\} \text{ statement-2 [; statement-3. . .]} \right]$$

NOTES:

- This form may only follow a STORE or RETRIEVE verb or a

$$\underline{\text{DELETE}} \text{ or } \underline{\text{HEAD}} \text{ or } \left\{ \begin{array}{c} \underline{\text{ADD}} \\ \underline{\text{SUBTRACT}} \\ \underline{\text{REPLACE}} \end{array} \right\} \text{ imperative statement.}$$

COMPATIBLES

IDS

- Statement-1 may only be a GO TO or a PERFORM imperative. Statement-2, statement-3, etc., may be any imperative statement appropriate to the basic verb, or a conditional of form 1 if appropriate.
- This form is provided to signal the occurrence of any logical or physical error as a result of some IDS command. The specific errors which may occur are a function of the command executed. A detailed coding scheme for identifying errors is defined in the GE-625/635 IDS Reference Manual (CPB-1093) and GE-400 Series IDS Reference Manual. The user program may determine the type of error by referring to a communication cell named ERROR-REFERENCE.

FILE AND DATA DESCRIPTIONS

As in COBOL, a clear distinction is made between the description of what processing is to be done (PROCEDURE DIVISION) and the characteristics of the data on which the processing is to be carried out (DATA DIVISION). The data description of IDS is formatted much the same as for COBOL and is supplementary to the normal DATA DIVISION of COBOL.

The file description entry provides information regarding the physical characteristics of the total IDS data file. The file description must be present as the first entry of the IDS SECTION. The entry consists of a level indicator, a file name, and a series of clauses which define the physical characteristics of the IDS file. The mnemonic level indicator MD is used to identify the start of the file description entry and to distinguish it from the record description entries which will follow.

IDS FILE DESCRIPTION COMPLETE ENTRY

FUNCTION: To furnish information concerning the physical structure of the IDS file.

Option 1.

MD file-name COPY FROM LIBRARY